

Brief Announcement: Self-Stabilizing Synchronization Of Arbitrary Digraphs In Presence Of Faults

Mahyar R. Malekpour

NASA Langley Research Center, Hampton, Virginia

Mahyar.R.Malekpour@NASA.GOV

1 Introduction

This brief announcement presents a fault-tolerant self-stabilizing distributed clock synchronization protocol for an arbitrary, non-partitioned digraph. Synchronization algorithms are essential for managing the use of resources and controlling communication in a distributed system. **Synchronization** of a distributed system is the process of **achieving** and **maintaining** a bounded skew among independent local time clocks. A distributed system is said to be self-stabilizing if, from an arbitrary state, it is guaranteed to reach a legitimate state in a finite amount of time and remain in a legitimate state. For clock synchronization, a legitimate state is a state where all parts in the system are in synchrony. The self-stabilizing distributed-system clock synchronization problem is, therefore, to develop an algorithm (i.e., a protocol) to *achieve* and *maintain* synchrony of local clocks in a distributed system after experiencing system-wide disruptions in the presence of network element imperfections. The **convergence** and **closure** properties address achieving and maintaining network synchrony, respectively.

The main challenge associated with distributed synchronization is the complexity of developing a correct and verifiable solution. It is possible to have a solution that is hard to prove or refute. Such a solution, however, is not likely to be accepted or used in practical systems. Thus, a proposed solution must be proven to be correct. The proposed solution must restore synchrony and coordinated operations after experiencing system-wide disruptions in the presence of network element imperfections and, for ultra-reliable distributed systems, in the presence of various faults. A fault is a defect or flaw in a system component resulting in an incorrect state [1]. Furthermore, addressing network element imperfections, e.g., oscillators drift with respect to real time and differences in the lengths of the physical communication media, is necessary to make a solution applicable to realizable systems.

There exist many clock synchronization algorithms for special cases and restricted conditions. There are many solutions that are based on randomization and, therefore, are non-deterministic, e.g., the second protocol in [2]. There are many solutions that deal with the closure property [3] but either do not address convergence or provide an ad hoc solution [4] for initialization and integration, separately. Typically, the assumed topology is a regular graph such as a fully connected graph or a ring. These topologies do not necessarily correspond to practical applications or biological, social, or technical networks. Furthermore, the existing models and solutions do not solve

the general case of the distributed synchronization problem. Even when the solutions achieve synchrony, the time to achieve synchrony is very large for many of the solutions.

We have addressed all these issues in our proposed solution. We have developed and mechanically verified a deterministic fault-tolerant self-stabilizing distributed clock synchronization protocol for an arbitrary, non-partitioned, strongly connected directed graph (digraph) ranging from fully connected to 1-connected network while allowing for differences in the network elements and tolerating detectably bad faults. Using authentication and error detection techniques, it is possible to substantially reduce the effects of variety of faults in the system. Furthermore, the classical definition of a self-stabilizing algorithm assumes generally that either there are no faults in the system [5] or all faults are detectable. Thus, we restricted our solution to detectably bad faults. Our proposed protocol does not rely on assumptions about the initial state of the system, and no central clock or a centrally generated signal, pulse, or message is used. Nodes are anonymous, i.e., they do not have unique identities. There is no theoretical limit on the maximum number of participating nodes. The only constraint on the behavior of a node is that the interactions with other nodes are restricted to defined links and interfaces. The protocol deterministically converges within a time bound that is a linear function of the self-stabilization period. There is neither a central system clock nor an externally generated global pulse or message at the network level. The communication links and nodes can behave arbitrarily provided that eventually the system adheres to the protocol assumptions. For a complete technical report about our proposed solution, related literature and protocols, the reader is referred to [5, 6, 7].

2 How The Protocol Works

In this section we provide an intuitive description of the protocol behavior. Each node is driven by an independent, free-running local physical oscillator (i.e., the phase is not controlled in any way) and a logical-time clock (i.e., a counter), denoted *LocalTimer*, which locally keeps track of the passage of time and is driven by the local physical oscillator. The nodes communicate with each other by broadcasting **Sync** messages. Broadcast of a message by a node is realized by transmitting the message, at the same time, to all nodes that are directly connected to it. A node periodically undergoes a resynchronization process either when its *LocalTimer* times out or when it receives a *Sync* message. If it times out, it broadcasts a *Sync* message and so initiates a new round of a resynchronization process. However, since we are assuming detectably bad faults, when a node receives a *Sync* message, except during a predefined time interval, it accepts the *Sync* message and undergoes the resynchronization process where it resets its *LocalTimer* and relays the *Sync* message to others. This process continues until all nodes participate in the resynchronization process and converge to a guaranteed precision. The predefined time interval where the node ignores all incoming *Sync* messages, referred to as **ignore window**, provides

a means for the protocol to prevent the endless cycle of resynchronization processes triggered by the succession of *Sync* messages.

3 Protocol Verification

A bounded model of the protocol was mechanically verified for a subset of digraphs and modeling challenges of the protocol and the system were addressed [6]. The model checking effort was focused on verifying correctness of the bounded model of the protocol as well as confirmation of claims of determinism and linear convergence with respect to the self-stabilization period. In [7] we present a deductive proof of the correctness of the protocol as it applies to networks consisting of unidirectional and/or bidirectional links. The crux of the proof is to answer whether or not it is possible for a message to circulate within the network without dying out and whether or not it is possible for all nodes to transmit *Sync* messages without ever timing out, assuming the synchronization period is sufficiently large. As a result of our analysis and verification effort, we conjecture that the protocol solves the general case of this problem. In [5] we also presented several variations of the protocol and discussed that this synchronization protocol is indeed an emergent system.

4 References:

1. Torres-Pomales, W; Malekpour, M.R.; Miner, P.S.: *ROBUS-2: A fault-tolerant broadcast communication system*, NASA/TM-2005-213540, pp. 201, March 2005.
2. Dolev, S.; Welch, J.L.: *Self-Stabilizing Clock Synchronization in the Presence of Byzantine Faults*, Journal of the ACM, Vol.51, No. 5, pp. 780-799, September 2004.
3. Srikanth, T.K.; Toueg, S.: *Optimal clock synchronization*, Journal of the ACM, 34(3), pp. 626–645, July 1987.
4. Davies, D.; Wakerly, J.F.: *Synchronization and matching in redundant systems*, IEEE Transactions on Computers, 27(6), pp. 531-539, June 1978.
5. Malekpour, M.R.: *A Self-Stabilizing Synchronization Protocol For Arbitrary Digraphs*, The 17th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC), Pasadena, California, December 12-14, 2011.
6. Malekpour, M.R.: *Model Checking A Self-Stabilizing Synchronization Protocol For Arbitrary Digraphs*, the 31st Digital Avionics Systems Conference (DASC), Williamsburg, Virginia, October 14-18, 2012, *to appear*.
7. Malekpour, M.R.: *Correctness Proof Of A Self-Stabilizing Synchronization Protocol For Arbitrary Digraphs*, NASA/TM-2011-217184, pp. 31, October 2011.